前端工程化开发方案

app-proto 介绍

厦门 · 智能住宿前端研发团队 @liyang24

内容提纲

- 业务特点 & 前端开发的一些总结
- 前后端分离（Node层功能介绍）
- 前端开发的一些“约定”
- 项目构建、打包、部署 & 运维

业务特点

返回 周边门店

搜索

按创... 仅公海 全城

鼓浪屿悠闲小屋客栈

评级 H0 房间数 0

兆和路25之三

上月支付 金额 ¥0 间夜 0

上月消费 金额 ¥0 间夜 0

鸟巢花园客栈（曾厝垵店）

评级 H0 房间数 0

曾厝垵117号之二

帐号管理 新帐号审批

搜索

运营注册 用户ID: 1532

闵彬彬

凯维娜精品公寓酒店

签约门店

修改

运营注册 用户ID: 1531

王斌

米腾时尚酒店

签约门店

米腾时尚酒店

PMS门店

修改

帐号 订单 消息 门锁

门店信息

基本信息 房间列表 订单日志 房间日志

2016-10-17 16:31:02

房型

操作者 翟万斌

将功能<自定义客人密码失效时间>设置为<12:00>

2016-10-17 16:27:18

房型 豪华大床

操作者 翟万斌

门店ID 1885 创建房型：豪华大床, 房间号：[609, 706]

2016-10-17 16:25:58

房型

操作者 翟万斌

创建客栈：爱巢精品主题公寓（民兵园店）

叮当管家 | PM测试叮当客栈 | 点我有惊喜 不信你试试 | 房态 | 订单 | 统计 | 渠道 | 设置 | 超市 | MT-陈...

2016-10-16

搜索房型或房间号

	10-16 日 剩51间	今天 一 剩52间 3	10-18 二 剩53间 1	10-19 三 剩54间	10-20 四 剩52间	10-21 五 剩54间	10-22 六 剩54间	10-23 日 剩54间	10-24 一 剩54间 1	10-25 二 剩54间
监胖子主题房 101										
102		iOS同学正在疯狂测 上门散客 2			啦啦啦啦 上门散客					
103	aa 上门散客									
10333		iOS同学正. 时 上门散客			iOS同学正.. 上门散客					
104	王玄烨 上门散客									
105		iOS同学正. 时 上门散客								
106										
107										
108										
会议室（勿改） 1122										
硬件实验室	恍恍惚惚 时 上门散客									
801										
802										

已预订 | 已入住 | 已离店 | 安装有门锁 | 电池电量低 | 脏房 | 门锁失联 | 门锁反锁 | 今日房态 | 远期房态

怡居时尚宾馆	怡居时尚宾馆	郑志东	1523	[redacted]	[redacted]	运营 注册	修改 删除

一种运行于浏览器的软件

远程部署，运行时增量下载的GUI软件

- 项目类型特点明显：单页面工具类应用，复杂表单
- 项目多 & 开发周期短（开发周期在三周以内）
- 面向用户：产品、销售、运营及开发人员
- 面临多后端服务（多个后端团队）
- 前后端并行开发

前端开发的一些总结

避免“大而全”的重量级框架，一个框架真的满足不了所有的业务场景

但是项目多了，我们又不想重新走一遍“轮子”

前端开发应该自成体系，不愿意再折腾一遍后端的开发环境

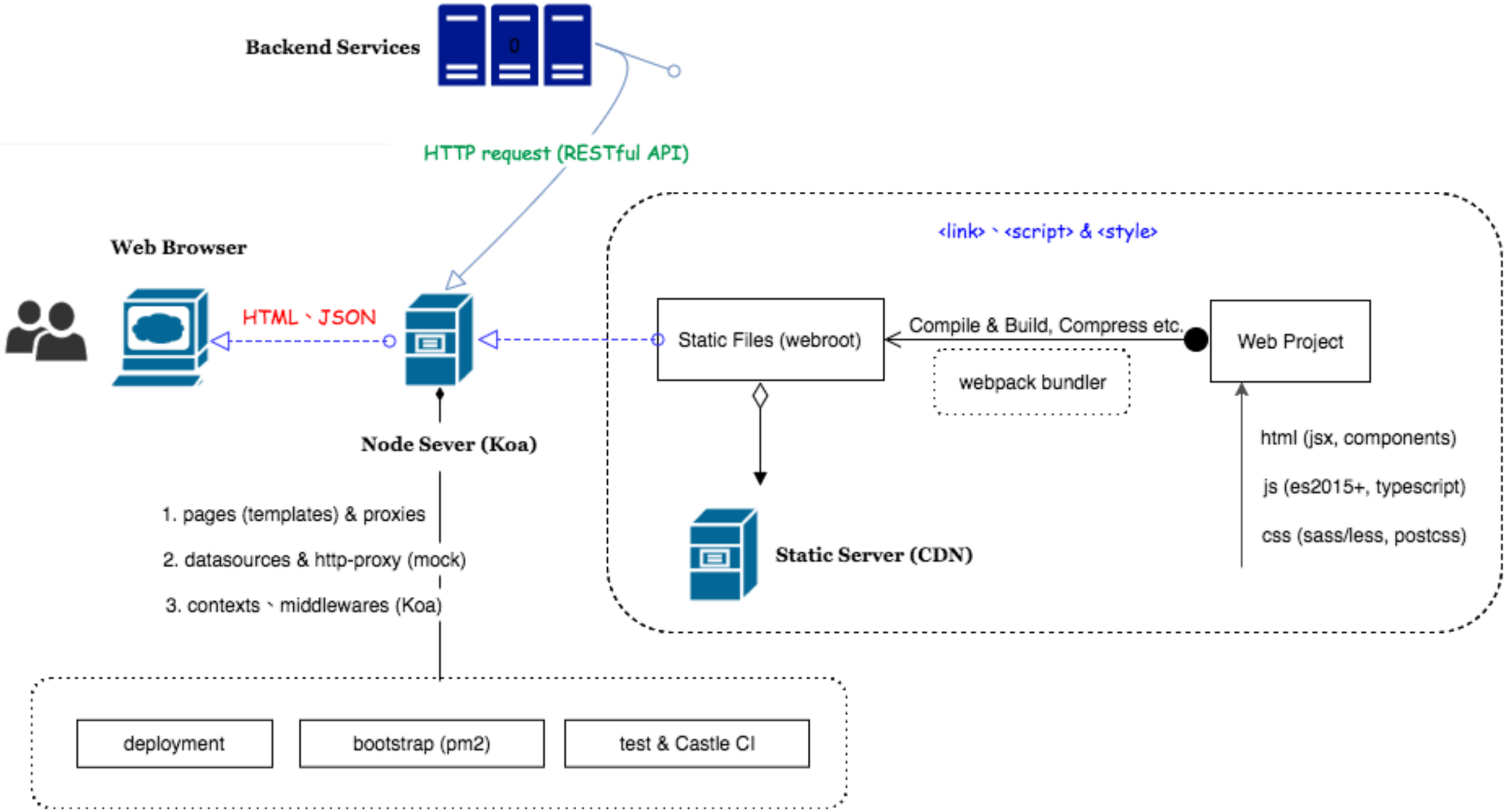
- 新技术和工具 不断涌现 & 迭代

- Vue/Vuex
- Angular2
- React/Redux

- Grunt/gulp
- webpack
- babel

合理的技术栈

- 合理分层、各层独立
- 任何一层可随时被替换、淘汰
- 框架不要做多余的事情


Backend Services
HTTP request (RESTful API)
Web Browser
HTML、JSON
Node Sever (Koa)
1. pages (templates) & proxies
2. datasources & http-proxy (mock)
3. contexts、middlewares (Koa)
deployment
bootstrap (pm2)
test & Castle CI
<link>、<script> & <style>
Static Files (webroot)
Compile & Build, Compress etc.
webpack bundler
Web Project
html (jsx, components)
js (es2015+, typescript)
css (sass/less, postcss)
Static Server (CDN)

约定优于配置

convention over configuration

前后端分离

- 前后端RD间的沟通成本降到最低
- 前端开发自成体系、更好地进行工程化
- 低运营成本

不意味着让前端工程师去承担后端工作，
而是让前端工程师能够更方便、更快捷地进行前端开发。

Node 层功能

数据代理 → HTTP 请求 → 从后端拿取数据、提供表单(RESTful API)

服务端路由 → 路由分发

- /user/login
- /perform/inbox
- /lst/paginate/1

服务端渲染 → 模板输出

- HTML 模板
- JSON

```
.
| |____config.js                          # 配置文件（对接后端各种测试环境）
| |____contexts                           # 自定义 Koa 上下文环境
| | |____http.js
| |____datasources                        # datasources
| | |____pms
| | | |____guest.js                       # 获取数据(RESTful API HTTP请求)
| | | |____guest.json                     # Mock 数据
| | | |____inn
| | | | |____create.js
| | | | |____create.json
| | | |____login.js
| | | |____login.json
| |____main.js
| |____middlewares                        # Koa 中间件
| | |____$global.js
| |____pages                              # 页面（Web端路由、渲染HTML）
| |____proxies                            # 代理（输出JSON）
| | |____index.js
| | |____mock                             # http-proxy 请求代理Mock目录
| | | |____pms
| | | | |____guest.json
| | | | |____login.GET.json
| |____templates                          # Node端渲染模板
|____server.js
```

数据中转

- datasources (将前端的Ajax请求放到Node层处理)
- http-proxy (传统方式)

按照约定写代码

后端接口(RESTful API)

```
pms/api/v2.01/login
pms/api/v2.01/inn/create
pms/api/v2.01/inn/get
upm/api/v3.15/menu
```

server/datasources/{后端系统}/{接口目录}

```
— datasources
    ├— pms
    |    ├— login.js
    |    ├— login.json
    |    └— inn
    |         ├— create.js
    |         └— get.js
    └— upm
         ├— menu.js
         └— menu.json
```

```
{ PmsLogin: [Function: bound method],
  PmsInnCreate: [Function: bound method],
  PmsInnGet: [Function: bound method],
  UpmMenu: [Function: bound method] }
```

`datasources/pms/inn/create.js` → `PmsInnCreate(params={}, mock=false)`

datasources/pms/login.js

```
/**
 * 用户登录
 **/
export default function async (params) {
  const http = this.http;
  const pms = this.config.api.pms;
  try {
    const apiUri = `${pms.prefix}/login`;
    const result = await http.post(apiUri, params);

    // 简单的数据格式检查
    if (Number(result.status) === 0 &&
        ('data' in result) &&
        ('bid' in result.data)) {
      // 将bid值记录至session
      this.session.bid = result.data.bid;
    }
    return result;
  } catch (e) {
    // 后端API出现异常（实时通知 or 记录日志）
  }

  return null;
}
```

- 没有跨域困扰（联调方便）
- 接口校验 & 二次加工

  （多后端服务，API格式不一定一致）
- 合并请求（避免前端同时发送多个Ajax请求）
- 缓存数据

  （如请求的城市字典、用户信息，短期内不会变动）
- HTTP Basic Authentication

datasources/pms/login.json

```
{
  "status": 0,
  "message": "成功",
  "data": {
    "bid": "@string(32)",
    "innCount": 1
  }
}
```

无后端接口？ mock支撑

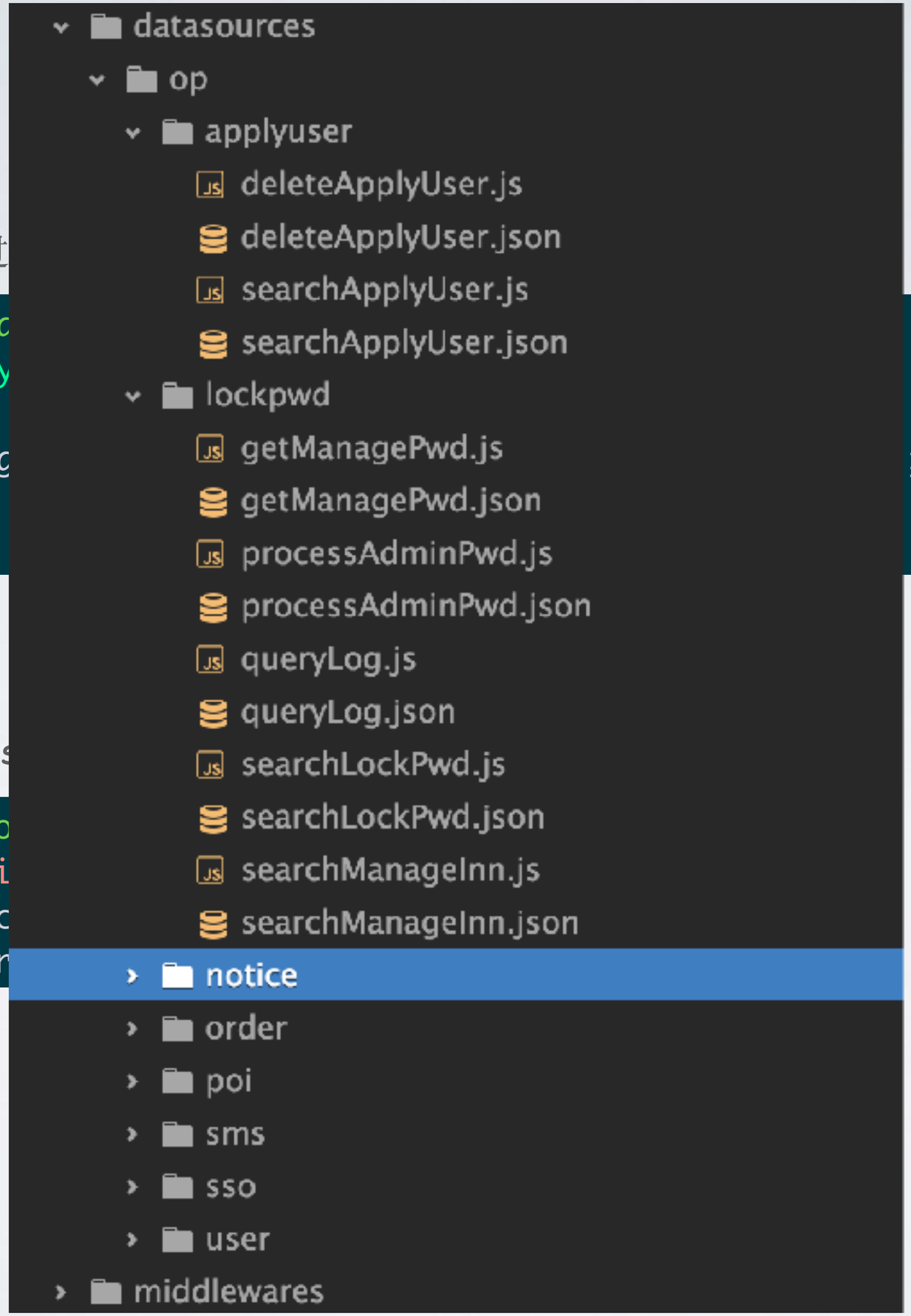
- http://mockjs.com/
- https://github.com/nuysoft/Mock/wiki/Syntax-Specification

Koa 中间件中通过

```
1 // Koa Midd
2 app.use(asy
3   // ...
4   const log
5   // ...
6 });
```

Web端统一封装ds

```
1 // Web (Bro
2 ds('PmsLogi
3   .then(suc
4   .catch(er
```



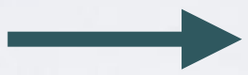
针对传统http-proxy实现的mock

Request	JSON File Path
GET /	index.GET.json
GET /shops	shops.GET.json
GET /shops/	shops/index.GET.json
POST /multiact/default	multiact/default.POST.json
GET /deal/*123456*	deal/:id.GET.json
GET /ktv/return/order?orderId=*123456*	ktv/return/order?orderId=:id.GET.json

https://github.com/meituan/monkey

服务端路由 & 渲染

```
— pages
    ├— index.js
    └— login.js
```



example.com/index

example.com/index/hello

example.com/login

example.com/login/error

- 配置服务端url路由：前端构建对应的js文件（约定）
- 服务端渲染功能：输出HTML template/JSON（提供服务端数据）

Schema模板约定

属性	类型	说明	默认值
url	*{array}*	路由正则。	以当前文件名创建***urls***，比如当前文件名为***index.js***，不填时会自动生成 ***['/index']*** 路由供使用。
methods	*{array}*	***HTTP Mehtods***	***['GET']***
js	*{array}*	需嵌入到***HTML template***中的js资源列表（一般通过此方式引入全局***js***库CDN地址）。	***[]***
css	*{array}*	全局***css***库***CDN***地址，和***js***功能相同。	***[]***
template	*{string}*	指定后端渲染的布局模板。	***default***
middlewares	*{array}*	针对本页面的***koa***中间件。	***[]***
controller	*{function*}*	数据调解；返回的可序列化对象可作为服务端数据。	要求必填。

server/pages/index.js

```
export default {
  urls: ['/pms'],         // 多种正则如：['/pms', ['/pms/v1'], ['/pms/v**']]
  methods: ['GET'],       // 多种method：['GET', 'POST']
  js: ['/assets/pms.js', 'http://code.jquery.com/jquery-1.12.0.min.js'],
  css: ['/assets/style.css'],
  template: 'default',
  middlewares: [],
  controller: function async(next) {
    return {foo: '来自服务端数据'};
  }
}
```

```
export default {
  urls: ['/pms'],
  controller: function async(next) {
    return {foo: '来自服务端数据'};
  }
}
```

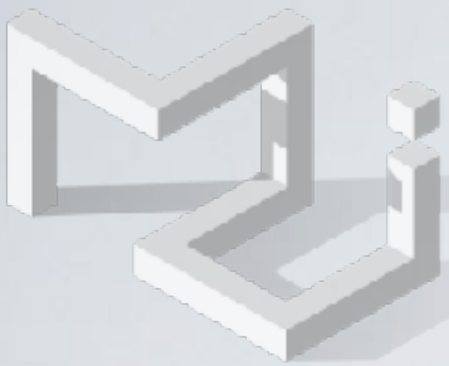
更简洁的写法

服务端渲染（Vue.js 示例）

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8"/>
        <title>app-proto</title>
        <script>window.serveData={}</script>
    </head>
   <body>
        <div id="app"></div>
        <script src="//cdn/file-5917b08e4c7569d461b1.js"></script>
   </body>
</html>

{
  "status":0,"message":"成功(POST)",
  "data":{"bid":"KsnL@^O5Qnw1C6p3#Z3sWCdty#bC^RE#","innCount":1}
}
```

Web端的一些约定



- Web 端的技术选型由前端工程师根据具体项目决策
- Ajax请求从Node端代理，而非具体后端服务
- 将JavaScript、CSS、HTML视为前端领域的“汇编”
- 组件化开发（复用）
  - 公共React、Angular组件集 (http://component.sankuai.com/)
  - 开源组件集：vux (WeUI)、ant.design、Material-UI

webpack

MODULE BUNDLER

BABEL

打包 & 构建

将JavaScript、CSS、HTML视为前端领域的“汇编”

- es2015+(babel.js)
- postcss (sass/less)
- JSX/Components

```
// 一年前的写法
var path = require('path')
// 当前的写法
const path = require('path')
// 未来的写法
import path from 'path'
```

严谨、复用（模块管理）


预编译
• es2015+语法转译、css预编译器处理
• git-hook & 源码质量审查
模块依赖解析
压缩+静态资源版本号
assets.json
静态资源映射表

生成的JS文件如何与Node 层的衔接: assets.json （资源表）

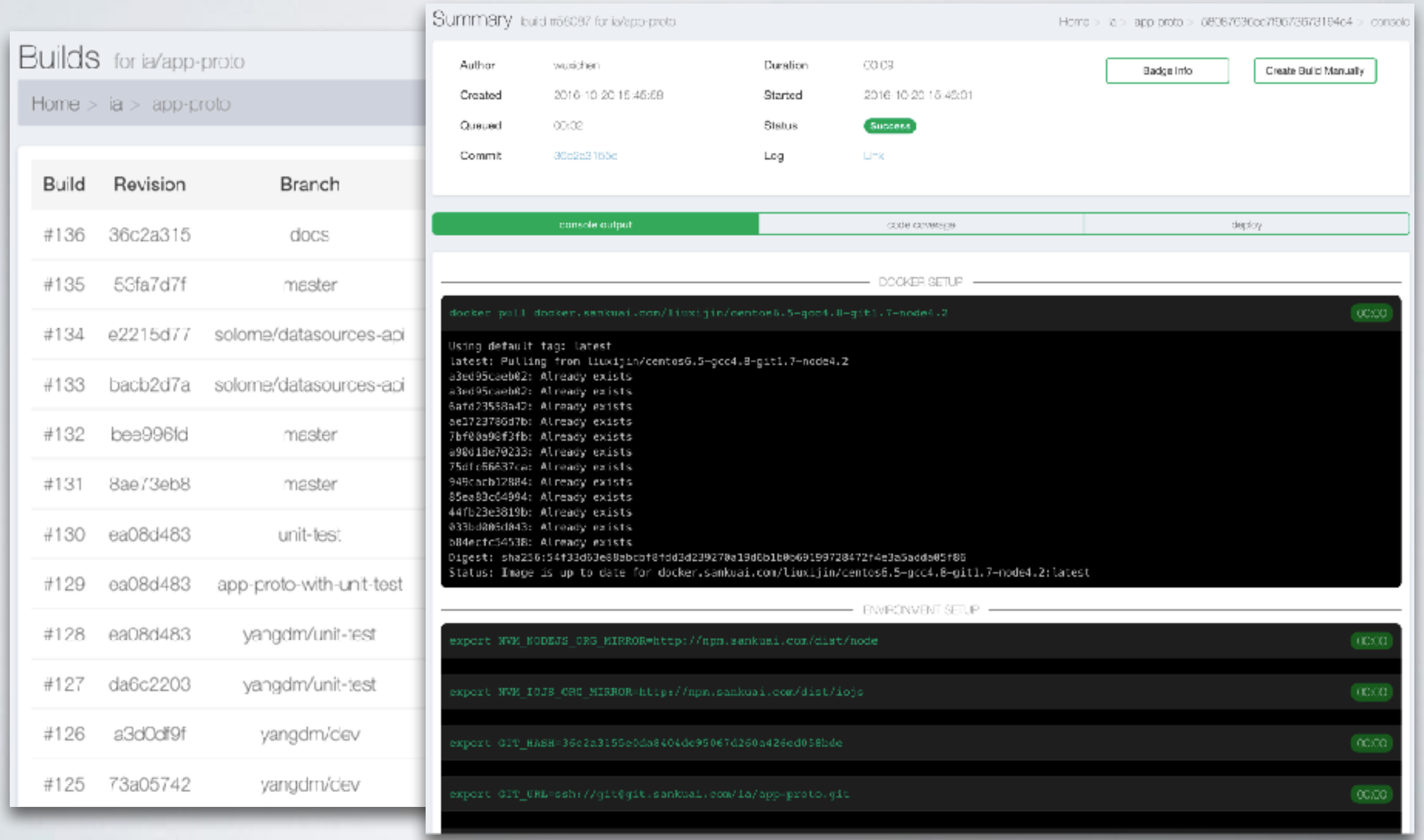
```
— pages
    ├— index.js
    └— login.js
```

+

```
{
  "index":
    {
      "js":"//s0.dingdanggj.net/pms/index-2abb99.js"
    },
  "login":
    {
      "js":"//s0.dingdanggj.net/pms/login-5917b0.js"
    }
}
```

```
<body>
  <div id="app"></div>
  <script src="//s0.dingdanggj.net/pms/index-2abb99.js"></script>
</body>
```

Castle CI 集成构建



脚手架

安装相关npm工具

```
npm install -g yo
mnpm install -g @ia/generator-app-proto@latest
```

创建项目目录

```
mkdir project-name
cd project-name
yo @ia/app-proto
```

执行 默认 DEMO

```
npm start
```

通过浏览器打开：http://localhost:8011

更多尝试

- 从项目中剥离 babel、webpack 依赖
- es2015+ => TypeScript 、CoffeeScript...


Questions?

謝

@liyang24

2016.10.22